



**STUD.IP**

3. Stud.IP-Entwickler-Workshop • 2. Juni 2006

**Workshop 3c:**

**Stud.IP-Enterprise-Edition**

André Noack, Frank Elsner

# Gliederung

- ▶ **Das Problem: Skalierbarkeit**
- ▶ **LAMP Tuning**
- ▶ **Mehr als ein Server**
- ▶ **Stud.IP und „shared nothing“**
- ▶ **In der Praxis: Osnabrück**
- ▶ **Diskussion**

# Das Problem: Skalierbarkeit

- ▶ Ein LAMP System mit Stud.IP ist schnell eingerichtet
- ▶ Ohne Last arbeitet es in jeder Konfiguration problemlos und unauffällig
- ▶ Armageddon: multiple chronologische Anmeldeverfahren, der Server steht
- ▶ Skalierbarkeit:
  - ▶ Die Fähigkeit, gestiegene Anforderungen kontrolliert und sicher anzunehmen

# LAMP Tuning

## ▶ Apache

- ▶ Apache Prozesse schlucken viel Speicher
- ▶ Mit mod\_php meistens bis ca. memory\_limit
- ▶ Die Prozesse schrumpfen nicht mehr, auch wenn sie zwischenzeitlich nur statischen Kontent liefern
- ▶ Standardeinstellungen für MaxClients grundsätzlich zu optimistisch
- ▶ Das Erzeugen neuer Prozesse wird nicht verhindert, auch wenn kein Speicher mehr vorhanden ist!

# LAMP Tuning

## ▶ Apache

- ▶ MaxClients vorsichtig wählen
- ▶ Ungefähr 2/3 des Speichers für Apache vorsehen, bei 1 GB wären das nur 34 (20MB pro Prozess) gleichzeitige Prozesse
- ▶ Dazu KeepAliveTimeout niedrig, da häufig alle Prozesse belegt sind
- ▶ PHPs memory\_limit so klein wie möglich setzen
- ▶ PHP Compiler Cache benutzen:
  - ▶ Zend, IonCube, APC, Turck, eAccelerator
  - ▶ Nicht nur schneller, sondern reduzieren auch den Speicherbedarf deutlich

# LAMP Tuning

## ▶ MySQL

- ▶ Query Cache benutzen (query\_cache\_size=20M)
- ▶ Key\_buffer\_size und table\_cache anpassen
  - ▶ key\_buffer\_size ca. größte Indexdatei der DB (besser alle Indexdateien)
  - ▶ table\_cache für Stud.IP 256
  - ▶ Kontrollieren mit MysqlAdmin
- ▶ Genug freien Speicher zum Caching übriglassen, Mysam Tabellen sind darauf angewiesen

# Mehr als ein Server

- ▶ **Evtl. leistungstärkeren Server benutzen**
  - ▶ Hardware ist gar nicht so teuer
  - ▶ Die Einrichtung und Pflege von mehreren kleinen Servern kostet auch
- ▶ **Einfach: Webserver und DB trennen**
  - ▶ Konfigurationen anpassen
  - ▶ Dedizierte Netzwerkverbindung
  - ▶ Problem: Zwei mögliche Ausfälle

# Mehr als ein Server

## ▶ Mehrere Webserver

- ▶ Kontent (Dateien) sollte ausgelagert werden, da es sonst repliziert werden muss
- ▶ Kann auch auf der DB Maschine unterkommen
- ▶ Dann können einfach zusätzliche Webserver hinzugeschaltet werden
- ▶ Lastverteilung muss geregelt werden

# Mehr als ein Server

## ▶ Mehrere DB Server

- ▶ Verkompliziert die Sache nicht unerheblich
- ▶ Webserverkontent sollte dann auf einer weiteren Maschine liegen (NAS)
- ▶ Replikation erfordert u.U. zusätzliche Anwendungslogik (Master-Slave Replikation, Schreibende Zugriffe nur auf dem Master)
- ▶ Andere Variante Master-Master, 2 Webserver die jeweils auf ihre eigene DB zugreifen

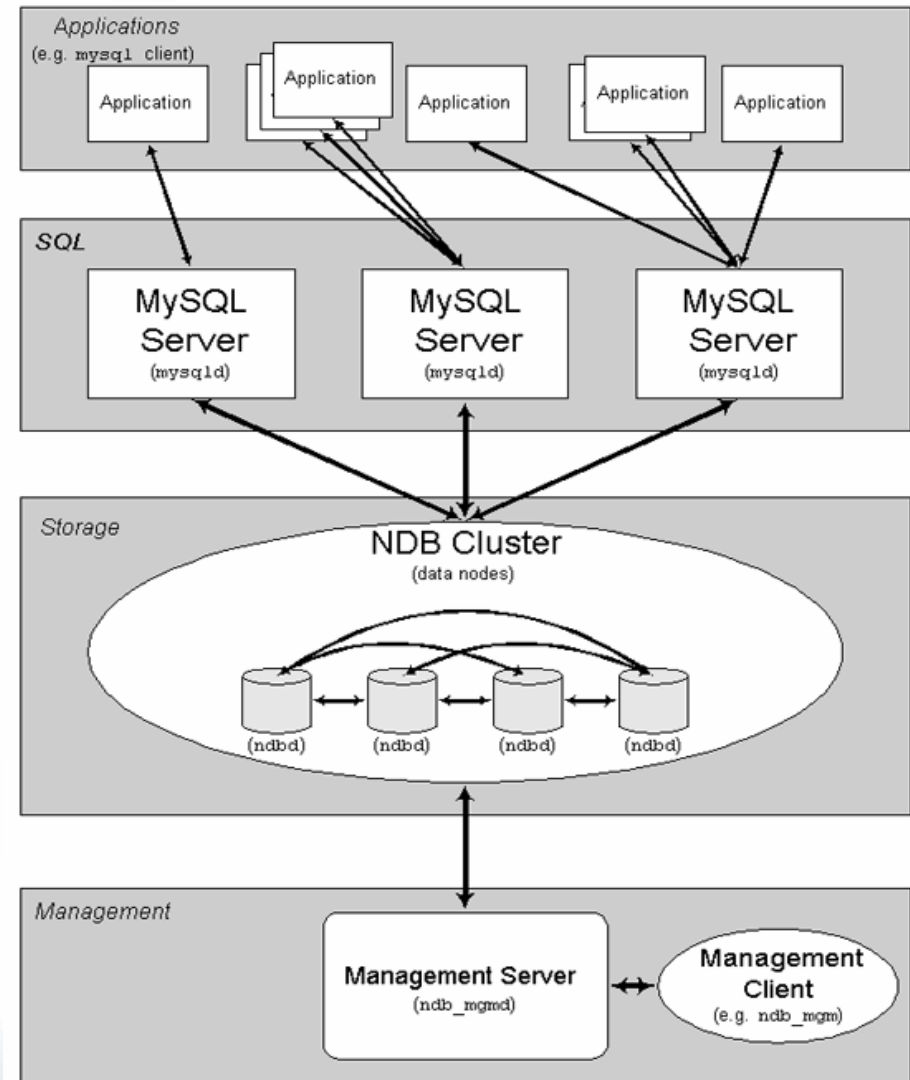
# Mehr als ein Server

## ▶ Mehrere DB Server

- ▶ Replikationsring sehr aufwändig, erfordert evtl. zusätzliche Lastverteilung der DBs, oder gleiche Anzahl an Webservern und DBs
- ▶ Mögliche Alternative: MySQL Cluster
  - ▶ Daten liegen in einer beliebigen Anzahl von data-nodes
  - ▶ Eine beliebige Anzahl von sql-nodes greift transparent auf die data-nodes zu
  - ▶ Der Ausfall eines data-nodes oder sql-nodes spielt keine Rolle
  - ▶ Erfordert einen zusätzlichen management-node

# MySQL Cluster

- ▶ min 2 data nodes, 1 management node
- ▶ sql nodes können direkt auf den Webservern laufen
- ▶ Also: 5 Maschinen!
- ▶ In 5.0x muss die DB komplett im Speicher gehalten werden, daher evtl. noch mehr data-nodes
- ▶ Wenn es denn mal läuft, fast beliebig skalierbar
- ▶ Konkrete Performance unsicher



# Stud.IP und „shared nothing“

- ▶ **Was ist „shared nothing“?**
  - ▶ Eine PHP (Web) Applikation sollte nicht selbst ihren Zustand verwalten
  - ▶ Diese Dinge sollten ausschließlich in der Datenschicht (DBMS) passieren
  - ▶ Daher:
    - ▶ Beliebige Skalierbarkeit (solange die Datenschicht mitspielt)
    - ▶ Transparente Ausfallübernahme

# Stud.IP und „shared nothing“

## ▶ Ist Stud.IP „shared nothing“?

### ▶ Nicht ganz:

- ▶ Die Datenschicht ist aufgeteilt auf das DBMS und das Dateisystem (Dateiuploads, temporäre Dateien, usw.)

- ▶ Chat benutzt shared memory

### ▶ Einfachste Lösung:

- ▶ Chat auf DB umstellen

- ▶ Dateiablage auf NAS/SAN legen

- ▶ Der Einfachheit halber alle Stud.IP Dateien auf NAS/SAN legen und über eine Alias Direktive verfügbar machen

# In der Praxis: Osnabrück

- ▶ Trennung Webserver Datenbank
- ▶ Zwei Webserver
- ▶ Statischer Content getrennt von dynamischem
- ▶ Compiler Cache: eAccelerator
- ▶ Volatile Daten auf SAN
- ▶ Experimente mit MySQL Cluster

# Diskussion

- ▶ Bis hierhin: Danke für die Aufmerksamkeit!
- ▶ Eigene Erfahrungen, Absichten
- ▶ Was bedeutet das alles für Entwickler?
- ▶ Engere Zusammenarbeit von Entwicklern und Betreibern (Administratoren)?
- ▶ Neue Features aus anderer Sichtweise betrachten
- ▶ Testen mit real world Daten